DEEP CONTEXTUALIZED WORD REPRESENTATIONS

or

ELMo

(Embeddings from Language Models)

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, Luke Zettlemoyer

Presented by Xiaoyan Wang (xiaoyan5@illinois.edu)
BACKGROUND

- Using pre-trained dense vectors as words representations has been a standard approach in NLP architectures
- They are lower in dimensionality (comparing to one-hot vectors)
- Can capture syntactic and semantic information
 Previous embedding methods (e.g. word2vec, GloVe) learn one vector representation for each word.
BACKGROUND

- However, words could have multiple different senses depending on their context, e.g.,

  "I went fishing for some sea bass"

- The bass line of the song is too weak"

- Contextual information is necessary in order to capture the semantic meaning of the word correctly
ELMo

(Embeddings from Language Models)
INTRODUCTION

- ELMo word representations are functions of the entire input sentence, i.e.
  - Previous methods: static embeddings (lexicon lookup)
  - ELMo: context-dependent embeddings (produced by a bi-directional language model)
ARCHITECTURE

- ELMo consists of $L$ layers of bi-directional language models.
- Input tokens are processed by a character-level CNN.
- Different layers of ELMo capture different information, so the final token embeddings should be computed as weighted sums across all layers.

The figure is taken from J Devlin et al., “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding.”
The raw texts are fed into a character-level convolutional neural network.

It uses 2048 character n-gram convolutional filters, two highway layers, and a linear projection down to 512-dimension.

Advantage: There will be no unknown words.

The figure is taken from Petr Lorenc’s blog, “ELMo”.

2048 vector for one word.
BIDIRECTIONAL LANGUAGE MODEL (forward)

- Given a sequence of $N$ tokens, $(t_1, t_2, \ldots, t_N)$,

- A forward language model computes

$$p(t_1, t_2, \ldots, t_N) = \prod_{k=1}^{N} p(t_k \mid t_1, t_2, \ldots, t_{k-1})$$

The figure is taken from Jay Alammar’s blog, “The Illustrated BERT, ELMo, and co. (How NLP Cracked Transfer Learning)”
BIDIRECTIONAL LANGUAGE MODEL (forward)

- Or, implementing as an $L$-layer LSTM:

$$p(t_1, t_2, \ldots, t_N) = \prod_{k=1}^{N} p(t_k \mid t_1, t_2, \ldots, t_{k-1}; \Theta_x, \Theta_{LSTM}, \Theta_s)$$

- Output of each LSTM is a context-dependent representation $\vec{h}_{k,j}$, where $k = 1, \ldots, N$ and $j = 1, \ldots, L$

The figure is taken from Jay Alammar’s blog, “The Illustrated BERT, ELMo, and co. (How NLP Cracked Transfer Learning)”
BIDIRECTIONAL LANGUAGE MODEL (backward)

• The backward language mode is implemented analogously

\[
p(t_1, t_2, \ldots, t_N) = \prod_{k=1}^{N} p(t_k \mid t_{k+1}, t_{k+2}, \ldots, t_N; \Theta_x, \Theta_{LSTM}, \Theta_s)
\]

• Output of each LSTM is a context-dependent representation \( \tilde{h}_{k,j} \),
  where \( k = 1,\ldots,N \) and \( j = 1,\ldots,L \), given \( (t_{k+1}, t_{k+1}, \ldots, t_N) \)

The figure is taken from Jay Alammar’s blog, “The Illustrated BERT, ELMo, and co. (How NLP Cracked Transfer Learning)”
BIDIRECTIONAL LANGUAGE MODEL (biLM)

- The forward and backward LM can be jointly maximizes

$$\sum_{k=1}^{N} \left( \log p(t_k \mid t_1, t_2, \ldots, t_{k-1}; \Theta_x, \Theta_{LSTM}, \Theta_s) + \log p(t_k \mid t_{k+1}, t_{k+2}, \ldots, t_N; \Theta_x, \Theta_{LSTM}, \Theta_s) \right)$$

- The params for token representation ($\Theta_x$) and Softmax layer ($\Theta_s$) are shared

The figure is taken from Jay Alammar’s blog, “The Illustrated BERT, ELMo, and co. (How NLP Cracked Transfer Learning)”
For each token $t_k$, the $L$-layer biLM computes $2L + 1$ vector representations

$$R_k = \left\{ h_{k,j} \mid j = 0, \ldots, L \right\}$$

where $h_{k,0}$ is the token layer and $h_{k,j} = [\vec{h}_{k,j}, \vec{h}_{k,j}]$ for each biLM layer.

To use in downstream model, ELMo collapses $R$ into a single vector

$$\text{ELMo}_{k}^{\text{task}} = E(R_k; \Theta^{\text{task}}) = \gamma^{\text{task}} \sum_{j=0}^{L} s_j^{\text{task}} h_{k,j}$$

The figure is taken from Jay Alammar’s blog, “The Illustrated BERT, ELMo, and co. (How NLP Cracked Transfer Learning)”
Usage

- Once pre-trained, we can freeze the weights of the biLM and use it to compute $\text{ELMo}_{k}^{\text{task}}$.

- Fine-tuning the biLM on domain specific data can lead to significant drops in perplexity increases in task performance.

- In general, ELMo embeddings $\text{ELMo}_{k}^{\text{task}}$ should be used in addition to a context-independent embedding $x_k$.

- Adding a moderate amount of dropout and regularize ELMo weights by adding $\lambda \| w \|_2^2$ to the loss.
RESULTS

- Pre-trained 2-layered ELMo on 1 Billion Word Benchmark (approximately 800M tokens of news crawl data from WMT 2011)

- The addition of ELMo increases the performance on various NLP tasks
  - question answering (SQuAD)
  - entailment/natural language inference (SNLI)
  - semantic role labeling (SRL)
  - coreference resolution (Coref)
  - named entity recognition (NER)
  - sentiment analysis (SST-5)

<table>
<thead>
<tr>
<th>TASK</th>
<th>PREVIOUS SOTA</th>
<th>OUR BASELINE</th>
<th>ELMO + BASELINE</th>
<th>INCREASE (ABSOLUTE/RELATIVE)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SQuAD</td>
<td>Liu et al. (2017)</td>
<td>84.4</td>
<td>81.1</td>
<td>85.8</td>
</tr>
<tr>
<td>SNLI</td>
<td>Chen et al. (2017)</td>
<td>88.6</td>
<td>88.0</td>
<td>88.7 ± 0.17</td>
</tr>
<tr>
<td>SRL</td>
<td>He et al. (2017)</td>
<td>81.7</td>
<td>81.4</td>
<td>84.6</td>
</tr>
<tr>
<td>Coref</td>
<td>Lee et al. (2017)</td>
<td>67.2</td>
<td>67.2</td>
<td>70.4</td>
</tr>
<tr>
<td>NER</td>
<td>Peters et al. (2017)</td>
<td>91.93 ± 0.19</td>
<td>90.15</td>
<td>92.22 ± 0.10</td>
</tr>
<tr>
<td>SST-5</td>
<td>McCann et al. (2017)</td>
<td>53.7</td>
<td>51.4</td>
<td>54.7 ± 0.5</td>
</tr>
</tbody>
</table>
ANALYSIS (Weighting Schemes)

- Different weighting schemes affect the performance on downstream tasks
- A large regularization parameter ($\lambda = 1$) reduce the weighting function to a simple average over the layers
- A smaller regularization ($\lambda = 0.001$) allows layer weights to vary

<table>
<thead>
<tr>
<th>Task</th>
<th>Baseline</th>
<th>Last Only</th>
<th>All layers</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>$\lambda=1$</td>
<td>$\lambda=0.001$</td>
</tr>
<tr>
<td>SQuAD</td>
<td>80.8</td>
<td>85.0</td>
<td>85.2</td>
</tr>
<tr>
<td>SNLI</td>
<td>88.1</td>
<td>89.3</td>
<td>89.5</td>
</tr>
<tr>
<td>SRL</td>
<td>81.6</td>
<td>84.6</td>
<td>84.8</td>
</tr>
</tbody>
</table>
ANALYSIS (Information Captured)

- Different layers of ELMo captured different information of the sentences
- For a 2-layered BiLM, the first layer captures **syntactic** information better and leads to a higher performance on POS tagging task
- The second layer captures **semantic** information better and leads to a higher performance on word sense disambiguation task

<table>
<thead>
<tr>
<th>Model</th>
<th>Acc.</th>
<th>Model</th>
<th>F₁</th>
</tr>
</thead>
<tbody>
<tr>
<td>Collobert et al. (2011)</td>
<td>97.3</td>
<td>WordNet 1st Sense Baseline</td>
<td>65.9</td>
</tr>
<tr>
<td>Ma and Hovy (2016)</td>
<td>97.6</td>
<td>Raganato et al. (2017a)</td>
<td>69.9</td>
</tr>
<tr>
<td>Ling et al. (2015)</td>
<td>97.8</td>
<td>Iacobacci et al. (2016)</td>
<td>70.1</td>
</tr>
<tr>
<td>CoVe, First Layer</td>
<td>93.3</td>
<td>CoVe, First Layer</td>
<td>59.4</td>
</tr>
<tr>
<td>CoVe, Second Layer</td>
<td>92.8</td>
<td>CoVe, Second Layer</td>
<td>64.7</td>
</tr>
<tr>
<td>biLM, First Layer</td>
<td>97.3</td>
<td>biLM, First layer</td>
<td>67.4</td>
</tr>
<tr>
<td>biLM, Second Layer</td>
<td>96.8</td>
<td>biLM, Second layer</td>
<td>69.0</td>
</tr>
</tbody>
</table>

Performance on POS tagging task on PTB

Performance on word sense disambiguation task (WSD)
ANALYSIS (Sample Efficiency)

- Adding ELMo to a model increases the sample efficiency and allow it to reaches higher performance with fewer data.
CONCLUSION

• ELMo produces contextualize word embeddings given a sentence

• The addition of ELMo embeddings improves the performance on a broad range of NLP tasks

• Different layers of ELMo encode different types of syntactic and semantic information
THANK YOU :}